Constraint Satisfaction Problems (CSPs)

A *binary constraint satisfaction problem* consists of

- A set of n variables $\{x_1, x_2, \dots, x_n\}$ with respective *finite* domains $D_1, D_2, \dots, D_n$
  - let $D = D_1 \cup D_2 \cup \dots \cup D_n$
  - let d be the size of the largest domain
- A set of e binary constraints $\{C_{ij}\}$
  - C_{ij} represents a constraint between variables x_i and x_j specifying the set of legal pairs of values
  - assume that $C_{ij}(u, v) = C_{ji}(v, u)$

Constraint graph

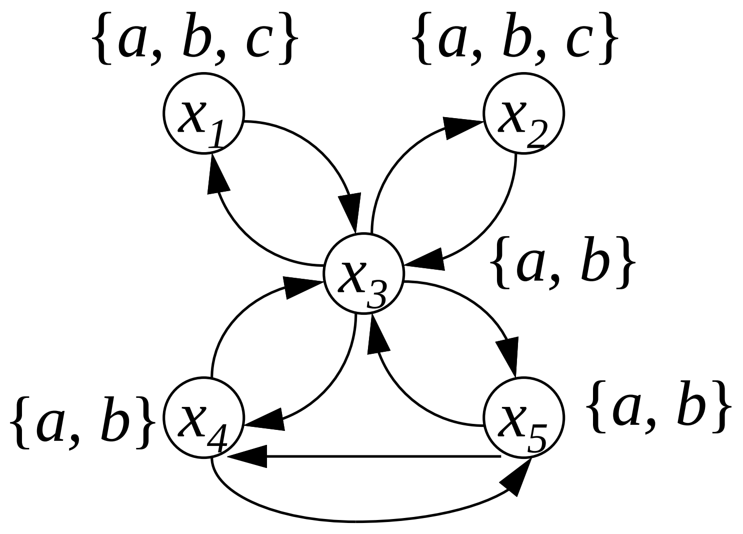
A constraint graph is a directed graph with *n* nodes and *e* edges

- Each variable is a node
- Each constraint C_{ij} is an edge from node x_i to node x_j

Variables $\{x_1, x_2, x_3, x_4, x_5\}$

Constraints

- $C_{31} = \{(a,b), (a,c), (b,c)\}$
- $C_{32} = \{(a,b), (a,c), (b,c)\}$
- $C_{34} = \{(a,b), (a,c), (b,c)\}$
- $C_{35} = \{(a,b), (a,c), (b,c)\}$
- $C_{54} = \{(a,b), (a,c), (b,c)\}$



Backtrack search

```
procedure $bcssp(n)$
    $consistent = true$
    $i = initialize()$
    loop
        if $consistent$ then $(i, consistent) = label(i)$
        else $(i, consistent) = unlabel(i)$
        if $i > n$ then return “solution found”
        else if $i = 0$ then return “no solution”
    endloop
end $bcssp$
```

Chronological backtracking: *initialize*

```
function initialize()
    for $i = 1$ to $n$
        $CD_i = D_i$                    /* initialize current domains  */
    endfor
    return $1$                          /* return the first variable */
end initialize
```

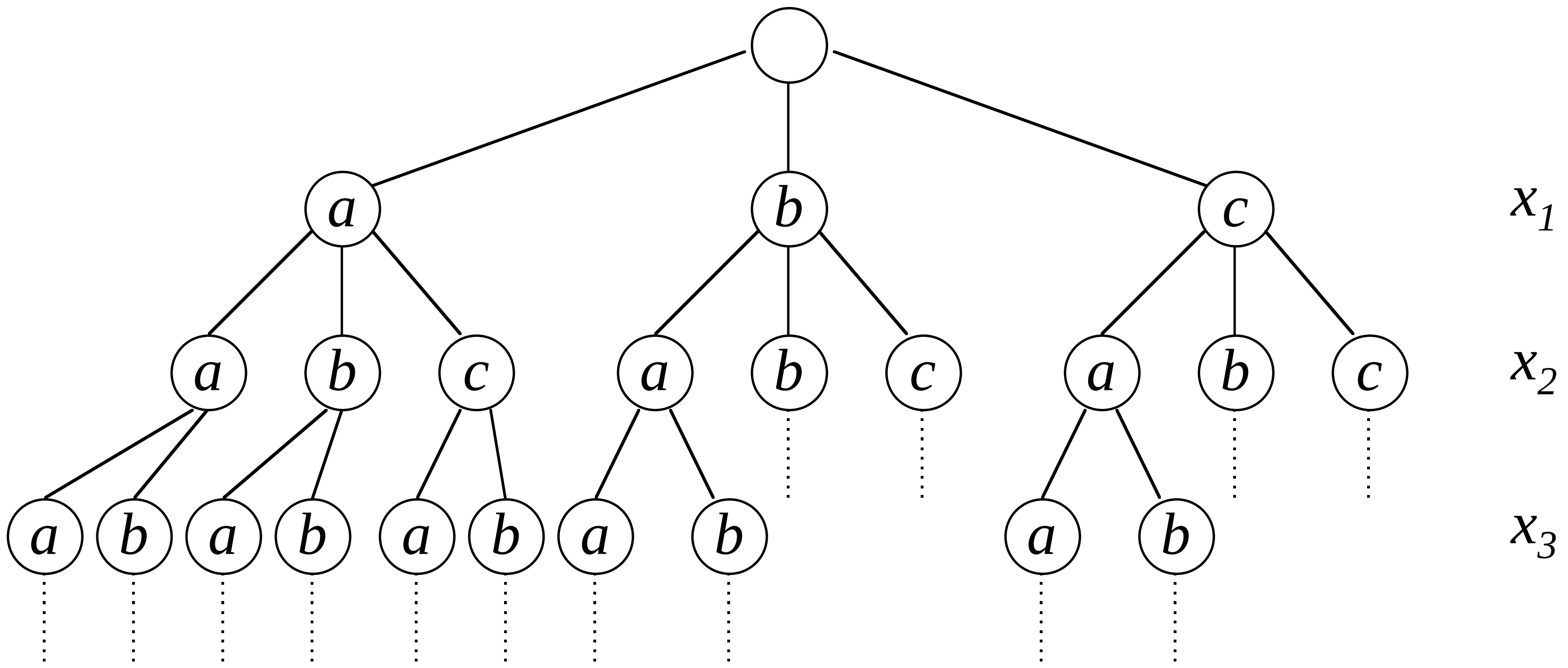
Chronological backtracking: *label*

```
function bt-label(i)
    for each $v_i \in CD_i$ do
        Set $x_i = v_i$ and $consistent = true$
        for each $x_j$ that has been previously assigned do
            if $\neg C_{ij}(x_i, x_j)$ then
                Remove $v_i$ from $CD_i$ and set $consistent = false$
                Unassign $x_i$ and break inner loop
            endif
        if consistent then return $(i+1, true)$
    endfor
    return $(i, false)$
```

Chronological backtracking: *unlabel*

```
function bt-unlabel(i)
    $h = i - 1$                /* Backtrack to previous variable */
    $CD_i = D_i$
    Remove current value assigned to $x_h$ from $CD_h$
    Unassign $x_h$
    if $CD_h$ is empty then
        return ($h$, false)
    else
        return ($h$, true)
end bt-unlabel
```

Example



Arc consistency

- An arc (i, j) in a constraint graph G is *arc consistent* with respect to domains D_i and D_j iff

  $\forall\, v \in D_i,\ \exists\, w \in D_j : C_{ij}(v, w)$

  - A graph G is arc consistent iff all its arcs are arc consistent

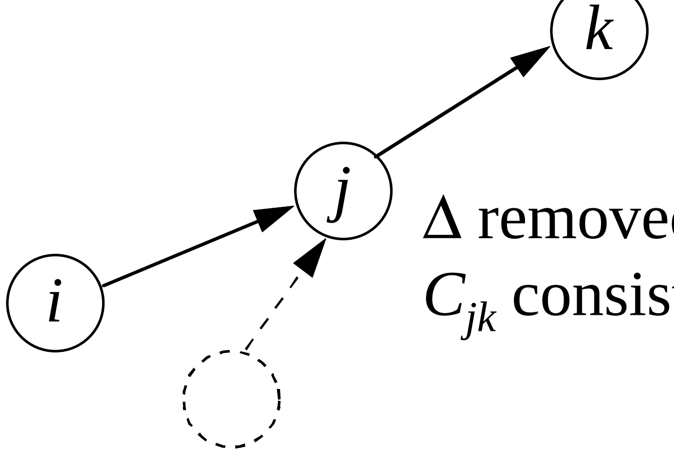
- Let $P = D_1 \times D_2 \times \ldots \times D_n$ and $P = D_1 \times D_2 \times \ldots \times D_n$ s.t. $P \supseteq P$. P is *the largest arc consistent domain for G in P* iff
  - G is arc consistent wrt P
  - there is no P such that $P \supseteq P \supset P$ and G is arc consistent wrt P
- **Theorem**: The largest arc consistent domain exists and is unique

AC-5

- AC-5 is a *generic* arc consistency algorithm
  - uses two abstract procedures *ArcCons* and *LocalArcCons*
  - can be specialized to either AC-3 or AC-4
  - can be specialized to exploit properties of constraints (e.g., functional, anti-functional, monotonic constraints)

Queue elements in AC-5

- AC-5 maintains a *queue* of elements of the form $((i, j), w)$
  - (i, j) is an arc, and w is a value in D_j that has been removed justifying the need to reconsider arc (i, j)
  - *Enqueue*(j, Δ, Q) inserts all elements of the form $((i, j), w)$ onto the queue Q such that (i, j) is an arc and $w \in \Delta$



Δ removed from D_j in making C_{jk} consistent

ArcCons and *LocalArcCons*

function *ArcCons*(i, j)

Returns $\Delta = \{ v \in D_i \mid \forall u \in D_j \neg C_{ij}(v, u) \}$

- Removing elements in Δ from D_i makes (i, j) arc consistent

function *LocalArcCons*(i, j, w)

Assumes that w has been removed from D_j

Returns Δ such that $\Delta_2 \supseteq \Delta \supseteq \Delta_1$ where

$\Delta_1 = \{ v \in D_i \mid C_{ij}(v, w) \text{ and } \forall u \in D_j \neg C_{ij}(v, u) \}$

$\Delta_2 = \{ v \in D_i \mid \forall u \in D_j \neg C_{ij}(v, u) \}$

Arc consistency with AC-5

```
procedure $AC\text{-}5(G)$
    $InitQueue(Q)$
    for each $(i, j) \in arc(G)$ do
        $\Delta = ArcCons(i, j)$
        $Enqueue(i, \Delta, Q)$
        $Remove(\Delta, D_i)$
    endfor
    while not $EmptyQueue(Q)$ do
        $((i, j), w) = Dequeue(Q)$
        $\Delta = LocalArcCons(i, j, w)$
        $Enqueue(i, \Delta, Q)$
        $Remove(\Delta, D_i)$
    endwhile
end $AC\text{-}5$
```

Counting queue operations in AC-5

- Introduce the *Status* of (*edge*, *value*) pairs such that
  - *InitQueue* sets *Status*((k, i), v) = *present* if v in D_i
    = *rejected* otherwise
  - *Enqueue* sets the *Status* of each queued item to *suspended*
  - *Dequeue* sets the *Status* of dequeued item to *rejected*

- *AC-5*'s loops preserve the invariant that *Status*((k, i), v)

= present	iff	v in D_i
= suspended	iff	v not in D_i and ((k, i), v) on the Q
= rejected	iff	v not in D_i and ((k, i), v) not on the Q

$\Rightarrow$ *AC-5* enqueues and dequeues at most $O(ed)$ items

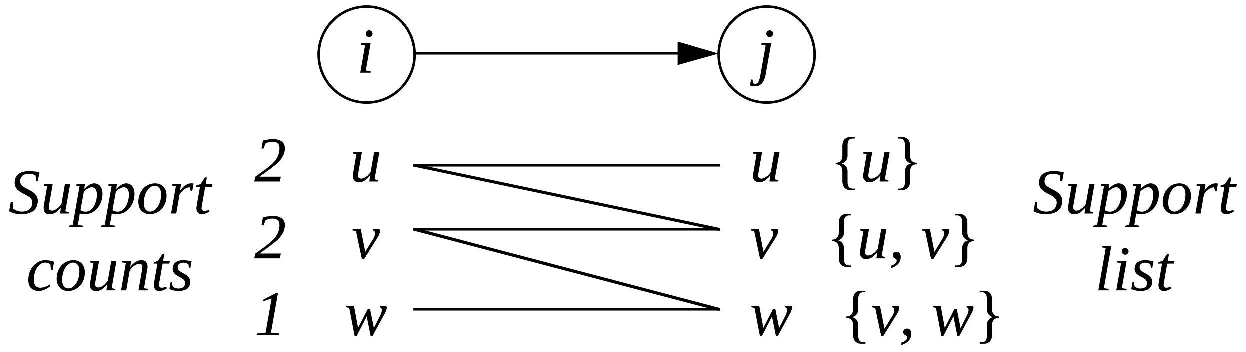
AC-3

- For arbitrary constraints *ArcCons* is $O(d^2)$
- *AC-3* is essentially *AC-5* in which *LocalArcCons* is implemented using *ArcCons*

$\Rightarrow$ *AC-3* is $O(ed^3)$

AC-4

- If *ArcCons* is $O(d^2)$ and *LocalArcCons* is $O(d)$then *AC-5* is $O(ed^2)$



Time per edge	$O(d^2)$	$O(d^2)$
Space per edge	$O(d)$	$O(d^2)$

- *LocalArcCons*(i, j, w) iterates through the "supports list" of w for edge (i, j), decrements "support counts", and computes Δ as the set of values whose "support counts" go to 0

⇒ *LocalArcCons* is $O(d)$ and *ArcCons* is $O(d^2)$ so *AC-4* is $O(ed^2)$

Functional constraints

- A constraint C is *functional* wrt a domain D iff for all $v \in D$ there exists at most one $w \in D$ such that $C(v, w)$

```
function ArcCons(i, j)
    Δ = {}
    for each $v \in D_i$ do
        if $f_{ij}(v) \notin D_j$ then $\Delta = \Delta \cup \{v\}$
    return Δ
end ArcCons

function LocalArcCons(i, j, w)
    if $f_{ji}(w)$  $D_i$ then return $\{f_{ji}(w)\}$
    else return {}
end LocalArcCons
```

ArcCons is $O(d)$

LocalArcCons is $O(1)$

AC-5 is $O(ed)$

Classes of constraints

- Other classes of constraints for which *AC-5* is $O(ed)$
  - anti-functional
  - monotonic
  - piecewise functional
  - piecewise anti-functional
  - piecewise monotonic